A computational framework for telemedicine

Ian Foster, Gregor von Laszewski, , George K. Thiruvathukal, Brian Toonen

Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439

Abstract

Emerging telemedicine applications require the ability to exploit diverse and geographically distributed resources. High-speed networks are used to integrate advanced visualization devices, sophisticated instruments, large databases, archival storage devices, PC's, workstations, and supercomputers. This form of telemedical environment is similar to *networked virtual supercomputers* also known as *metacomputer*. Metacomputers are already being used in many scientific application areas. In this article, we analyze requirements necessary for a telemedical computing infrastructure and compare them with requirements found in a typical metacomputing environment. We will show that metacomputing environments can be used to enable a more powerful and unified computational infrastructure for telemedicine. The Globus metacomputing toolkit can provide the necessary low level mechanisms to enable a large scale telemedical infrastructure. The Globus toolkit components are designed in a modular fashion and can be extended to support the specific requirements for telemedicine.

1 Introduction

In the last decade, dramatic advances in software and hardware technology have changed the landscape for telemedical computing. Today, personal computers have reached a performance level which one could have only dreamed of, a couple of years ago. The increase in processor speed is accompanied by the availability of memory to economical prices. New graphic cards let a PC perform at almost workstation speed. On the high end, vector-supercomputers are outperformed by distributed memory parallel architectures. Furthermore, changes in software engineering are marked by the acceptance of object oriented programming concepts and languages.

[1] E-mail contact of the authors: {gregor,thiruvat,itf,toonen}@mcs.anl.gov

Computer networks have evolved from local area networks to wide area networks. The new hardware enables the distribution of information in a global “World Wide Web”. The World Wide Web has established itself as a functioning computing environment, accessible by the ever-increasing number of online users. Starting from the desire to provide a mechanism for exchanging data between scientists, it has reached the potential to become the computing platform of the future. Hardware advances in the network technology, like the introduction of the ATM technology (Asynchronous Transfer Mode), provide the necessary backbone for the fast information exchange between computers.

Currently, the WWW is most frequently used for exchanging data, and allowing online users to access information stored at remote sites. Besides redistributing information, the WWW can be used to redistribute computations on different compute servers. Computations can be mapped to idle compute servers or unique compute resources which otherwise would not be available. An environment managing the resources of many cooperating diverse computers and peripherals is known as *metacomputer*. Without doubt, an environment build around a ”metacomputer,” will influence not only developments in telemedicine but also in other applications areas.

One of the goals of this paper it to examine the possibility to reuse existing metacomputing technologies for telemedicine. First, we will define the term telemedicine and derive necessary requirements that influence the design of a compute infrastructure. We show that this infrastructure is similar to existing metacomputing testbeds as used in the Globus metacomputing framework. We analyze the Globus metacomputing framework and indicate where it can help in order to support telemedical applications.

2 Telemedicine

Telemedicine is an emerging discipline, which utilizes the newest technologies to transmit medical information with the help of electric signals. From this broad definition, it is clear that the applications in telemedicine are of a wide spectrum [26,20,24,25,27,23]. Telemedicine is used in the following areas:

- Tele consulting and assistance
  - Remote supervision
  - Remote consulting
  - Medical Video Conferencing
- Virtual medical libraries
  - Document distribution via WWW
  - Medical databases
  - Research databases

- Virtual medical stores
  - Accounting
  - Product databases
  - Service Brokers

This list is not comprehensive but shows some examples where telemedicine is used currently. In particular, telemedicine is in wide practical use for teleconsulting between doctors in geographically disperse locations. Video conferencing is used for diagnosing, as well as, educational purposes. Today, material containing medical information is often distributed via the WWW. The functionality of the WWW in the area of telemedicine reaches from buying medical products to a visit of a virtual telemedical office. Telemedicine is successfully used to reach underserved areas. Some of the leading goals of telemedicine are to enable

- an increase of the availability of services,
- instant access to data,
- secure access and exchange of data,
- user friendly access,
- a high quality of service, and
- to reduce the cost for the health service.

To determine the requirements of a telemedical environment we analyze the simple scenario depicted in Figure 1. A patient is picked up by an ambulance, brought to the hospital and examined by a doctor with support of a nurse. Many interactions take place including the communication with a laboratory, as well as, the lookup of medical data in an external database collected by researchers. A unique aspect of a telemedical environment is that some medical equipment and other resources are mobile. In addition, the patient itself is exposed to different environments at different times. The scenario can be extended in many ways. It would be beneficial to equip the ambulance with a general position system (GPS) which inputs data to a traffic control and prediction system in order to find the fastest way to transport the patient to the hospital. Data gathered in the ambulance could be transferred on hand-held devices or via wireless communication devices to the hospital database. This rapid information exchange is essential for the decision process to invoke a proper treatment for the patient.

In order to build a telemedical environment for our previous scenario we need to include many different resources as categorized in Figure 2. *Hardware resources* allow to perform computations, measurements, and the visualization of results. Since they can be on different geographical locations, it must be possible to communicate between them via networks. *Software resources* allow the utilization of the hardware resources with the help of specialized programs. Very important components of a telemedical infrastructure are *knowledge re-*

sources. They include doctors, nurses, researchers, but also more abstract resources like libraries and databases. They are especially important since they build the heart of a functional health care environment.

The previous scenarios suggest a set of requirements for a telemedical computing infrastructure:

- *Multiple administrative domains.* The resources used in the telemedical environment are not owned or administered by a single entity. The need to deal with multiple administrative entities complicates the already challenging network security problem, as different entities may use different authentication mechanisms, authorization schemes, and access policies. The need to execute user-supplied code at different sites introduces additional concerns.
- *Heterogeneous community.* The users of a telemedical infrastructure have different requirements and goals which leads. The integration of a heterogeneous community in a telemedical structure poses many problems and can lead to conflicts. An example for such a conflict is the requirement to build a transparent system for the user while enabling maximum security for data access (see below).
- *Heterogeneity at multiple levels.* Both the computing resources used to construct the telemedical infrastructure and the networks that connect the resources are often heterogeneous. Heterogeneity arises at multiple levels, ranging from physical devices, through system software, to scheduling and usage policies.
- *Unpredictable structure.* Traditionally, high-performance computing applications have been developed for a single class of system with well-known characteristics—or even for one particular computer. In contrast, telemedicine requires executing diverse applications in a wide range of environments, constructed dynamically from available resources. Geographical distribution and complexity are other factors that make it difficult to determine system characteristics such as network bandwidth and latency *a priori.*
- *Dynamic and unpredictable behavior.* Traditional high-performance systems use scheduling disciplines such as space sharing or gang-scheduling to provide exclusive—and hence predictable—access to processors and networks. In telemedical environments, resources—especially networks—are more likely to be shared. One consequence of sharing is that behavior and performance can vary over time. For example, in wide area networks built using the Internet Protocol suite, network characteristics such as latency, bandwidth and jitter may vary as traffic is rerouted. Large-scale telemedical systems may also suffer from network and resource failures. In general, it is a difficult problem to guarantee even minimum quality of service requirements.
- *Transparency for users and patients.* Humans have a very important role in the telemedical environment. Many resources are used by doctors, nurses, researchers and healthcare providers. This variety of potential users of a telemedical environment results in different requirements for each of the

users. The requirements are often overlapping. Goal should be to simplify a task for a user of the telemedical environment leading to a transparent environment.

- *Need for an open environment.* The example to add a traffic control system based on a GPS is a prime example for the need that a telemedical environment needs to be able to incorporate other technologies in order to increase its functionality.
- *Need for Security.* Since the data stored in the databases include sensitive data special security mechanisms have to be in place to ensure the integrity of databases or other sources for information. Many administrative domains are likely to use firewalls. A telemedical infrastructure must have a way to include mechanisms dealing with the security.
- *Need to discover the state of the environment.* Fundamental to many issues as listed above is the need for mechanisms that allow telemedical applications to obtain real-time information about system structure and state. This information is used to make configuration decisions and to notify when information changes. Required information can include network activity, available network interfaces, processor characteristics, and authentication mechanisms. It also includes information about the availability and special knowledge of doctors, nurses and others, as well as their availability. Such interactions can be modeled with the help of complex networked systems. Decision processes require combinations of these data in order to achieve efficient end-to-end configuration of complex networked systems.
- *Scale and the need for selection.* Due to the extreme distributed nature of telemedical data and resources, it is important to develop a scalable telemedical infrastructure. Resource selection must be provided to allow the access of a subset of resources. Deadlock prevention and detection algorithms have to be included in resource selection and reservation strategies.
- *Need for resource scheduling.* Due to the high degree of variability in a telmedical environment sophisticated scheduling pollicies have to be supported with the help of specialized scheduling and resource allocation algorithms.
- *Need for quality of service.* It is critical to provide a Quality of Service (QoS) for many telemedical applications. While in some videoconferences a drop of frames might be expectable, a remote computational steering of a remote surgery should not be interrupted. This technology restriction will pose also restrictions to what is possible in a telemedical environment. Careful analysis of the tasks to be performed and their reliability has to be undertaken to not jeopardize the outcome of, for example, a remote operation.

Before we will go into more detail for each of the categories, we would like to introduce the Globus Metacomputing Toolkit. This way we can show more easily how a metacomputing toolkit can enable a telemedical environment on different abstraction levels.

3 The Globus Metacomputing Infrastructure Toolkit

Emerging telemedicine applications require the ability to exploit diverse and geographically distributed resources. High-speed networks are used to integrate advanced visualization devices, sophisticated instruments, large databases, archival storage devices, PC's, workstations, and supercomputers. This form of telemedical environment is similar to *networked virtual supercomputers* also known as *metacomputer*. Metacomputers are already being used in many scientific applications areas. Requirements necessary for a telemedicine infrastructure are similar to requirements found in a typical metacomputing environment. Thus, metacomputing environments can be used to enable a more powerful and unified computational infrastructure for telemedicine. The Globus metacomputing toolkit can provide the necessary low level mechanisms to enable a large scale telemedical infrastructure. The Globus toolkit components are designed in a modular fashion and can be extended to support the specific requirements for telemedicine.

A number of pioneering efforts have produced useful services for the metacomputing application developer. For example, Parallel Virtual Machine (PVM) [10] and the Message Passing Interface (MPI) [12] provide a machine-independent communication layer, Condor [14] provides a uniform view of processor resources, Legion [11] builds system components on a distributed object-oriented model, and the Andrew File System (AFS) [17] provides a uniform view of file resources. Each of these systems has been proven effective in large-scale application experiments.

Our goal in the Globus project is not to compete with these and other related efforts, but rather to provide basic infrastructure that can be used to construct portable, high-performance implementations of a range of such services. To this end, we focus on (a) the development of low-level mechanisms that can be used to implement higher-level services, and (b) techniques that allow those services to observe and guide the operation of these mechanisms. This approach reduces the complexity and improves the quality of metacomputing software by allowing a single low-level infrastructure to be used for many purposes, and by providing solutions to the configuration problem in metacomputing systems.

The Globus toolkit comprises a set of modules. Each module defines an *interface*, which higher-level services use to invoke that module's mechanisms, and provides an *implementation*, which uses appropriate low-level operations to implement these mechanisms in different environments. Currently identified toolkit modules are as follows:

- *Resource location and allocation.* This module provides mechanisms for expressing application resource requirements, for identifying resources that meet these requirements, and for scheduling resources once they have been located. Resource location mechanisms are required because applications generally cannot be expected to know the exact location of required resources, particularly when load and resource availability can vary. Resource allocation involves scheduling the resource and performing any initialization required for subsequent process creation, data access, etc. In some situations—for example, on some supercomputers—location and allocation must be performed in a single step.

- *Communications.* Basic communications mechanisms must permit the efficient implementation of a wide range of communication methods, including message passing, remote procedure call, distributed shared memory, stream-based, and multicast. Mechanisms must be cognizant of network quality of service parameters such as jitter, reliability, latency, and bandwidth.
- *Unified resource information service.* This component provides a uniform mechanism for obtaining real-time information about metasystem structure and status. The mechanism must allow components to post as well as receive information. Support for scoping and access control is also required.
- *Authentication interface.* The authentication interface module provides basic authentication mechanisms that can be used to validate the identity of both users and resources. These mechanisms provide building blocks for other security services such as authorization and data security that need to know the identity of parties engaged in an operation.
- *Process creation.* It is essential to be able to initiate computation on a resource once it has been located and allocated. This task includes setting up executables, creating an execution environment, starting an executable, passing arguments, integrating the new process into the rest of the computation, and managing termination and process shutdown.
- *Data access.* The data access module provides high-speed remote access to persistent storage such as files. Some data resources such as databases may be accessed via distributed database technology or the Common Object Request Broker Architecture (CORBA). The Globus data access module addresses the problem of achieving high performance when accessing parallel file systems and network-enabled I/O devices such as the High Performance Storage System (HPSS).

Together, the various Globus toolkit modules can be thought of as defining building blocks of a *metacomputing virtual machine.* The definition of this virtual machine simplifies application development and enhances portability by allowing programmers to think of geographically distributed, heterogeneous collections of resources as unified entities. Using this building blocks enables one to design a metacomputer, as been demonstrated at SC'97 with the Gusto testbed which had an overall computational power of about 2Tflops.

We now describe in more detail the Globus communications, information service, authentication, and data access services. In each case, we outline how the component maps to different implementations, is used to implement different higher-level services.

3.1 Communications

The success of telemedecine is largely dependent on an extensive network infrastructure. The network technologies used in a telemedical environment include asynchronous transfer mode (ATM), satellites, asymmetrical digital subscriber line (ADSL), and cable modems. Lower bandwidth technologies are standard analog telephone lines, and digital networks with integrated services digital network (ISDN). The proper selection of a communication service is determined by the type, the amount, and the urgency of the information exchange (Table 1). The choice is often determined by a cost-benefit factor. A compute environment must support such a cost function provided by the administrative domain. The Globus communication module can be used to implement the necessary infrastructure to allow to geographical disperse computers to communicate with each other.

The Globus communications module is based on the Nexus communication library [8]. Nexus defines five basic abstractions: nodes, contexts, threads, communication links, and remote service requests. The Nexus functions that manipulate these abstractions constitute the Globus communication interface. This interface is used extensively by other Globus modules and has been used to construct various higher-level services, including parallel programming tools. Active Messages [15] and Fast Messages [21] have similarities in goals and approach, but there are also significant differences [6]. The Nexus interface and implementation support rule-based selection of the methods—such as protocol, compression method, and quality of service—used to perform communication [6]. Different communication methods can be associated with different communication links, with selection rules determining which method should be used when a new link is established. These mechanisms have been used to support multiple communication protocols [6] and selective use of secure communication [7] in heterogeneous environments. The Nexus communication Library is capable to support most requirements that are posed by a telemedical infrastructure.

The rule based selection of communication protocols, compression methods and quality of service makes the Nexus library an ideal choice for a low level communication module. Higher level communication services are available via MPI a partial implementation of Nexus in Java. The Java-Nexus port allows that PC's can be integrated in a large-scale computational framework. Due to the amount of sensitive data transmitted over networks, it is essential to build a secure network infrastructure. Nexus allows using a key based security mechanism for the communication between different computational nodes.

3.2 Computations

A powerful communications library provides the basis for distributing the computational workload over many processors. The usage of today's supercomputers makes it possible to modify the traditional view-only utilization of expensive input devices (computed tomography) towards real time steering during an examination. This has the advantage that during an exam regions of interest can be explored immediately and a diagnosis can be achieved more quickly and accurately (Figure 3). Other examples for compute intense applications are the x-ray analysis of large molecular structures with the help of unique beamlines at synchrotrons, and the analysis of gene sequences [29,9]. Thus, the availability of immense computational power enabled by an integrated network infrastructure will allow to reduce the turnaround times for existing applications and to solve problem instantiations which were simply to big to be solved on a previous computational (telemedicine) infrastructures.

3.3 Metacomputing Directory Service

Metacomputing environments depend critically on access to information about the underlying networked supercomputing system. Required information can include configuration details about resources such as the amount of memory, CPU speed, number of nodes in a parallel computer, or the number and type of network interfaces available. Important are also instantaneous performance information, such as point-to-point network latency, available network bandwidth, and CPU load. Furthermore, application-specific information, such as memory requirements or program parameters found effective on previous runs have to be considered.

Different data items will have different scopes of interest and security requirements, but some information at least may potentially be required globally, by any Globus component. Information may be obtained from multiple sources: for example, from standard information services such as the Network Information Service (NIS) or Simple Network Management Protocol (SNMP); from specialized services such as the Network Weather Service [30]; or from external sources such as the system manager or an application.

Globus defines a single, unified access mechanism for this wide range of information, called the Metacomputing Directory Service (MDS) [5]. Building on the data representation and application programming interface defined by the Lightweight Directory Access Protocol (LDAP), MDS defines a framework in which can be represented information of interest in distributed computing applications.

This service can provide the basic information for static and dynamically changing resources in a telemedical infrastructure. The MDS cannot only be used to represent information about the compute environment, but it also addresses the problem of how to store other information in a uniform way across heterogeneous compute platforms. In a telemedicine environment information such information could include equipment which does not perform a compute function but rather a data acquisition function (e.g., expensive microscopes, imaging technology). MDS (and LDAP) could be used to store basic patient locator information as patients themselves are dynamically changing resources; however, detailled patient information belongs in a relational or object database, where issues such as security, data integrity, and transaction processing are addressed more adequately.

Each administrative domain can be responsible for its own LDAP server(s). This helps to simplify the issue of security and allow each domain be in control of its own security policies. It is straightforward to create a representation for patient data as a set of LDAP objectclass specifications. Nevertheless, better tools have to be developed to be able to enhance the object definition of objects to be stored in the LDAP tree. Work on active directories will lead to these enhancements in all likelihood.

As a concluding example of how MDS can be employed in telemedicine, the information about a patient can be stored in the MDS as a hybrid data object consisting of open and closed parts. The open part is needed for the patient to participate as a resource in the metacomputing (telemedicine) environment. The closed part is needed to find detailled (and secure) information about the patient. The closed part only contains a single item of information to locate the patient data, e.g. an object identifier, in a proprietary database. This pragmatic design consideration allows each administrative domain to manage its proprietary information yet allows other domains to access proprietary data objects using vendor-specific programming interfaces. Furthermore, the ability to interface to relational and object technology allows the best technology to be provided for the many consumers of information in the telemedicine environment. Relational database technology allows the definition of *views* to expose only the necessary and desired information to the outside world.

Special care has to be taken in order to evaluate if the database technology is supported on most of the relevant compute platforms for telemedicine. LDAP is supported on most of them ranging form PC's to Unix workstations to supercomputers. Since the source code is publicly available, a port to other platforms seems to be straightforward. In addition, LDAP is today available as a pure Java implementation, which will eliminate future compatibility issues.

3.4 Authentication Methods

The Globus authentication module supports password, Unix RSH, and Secure Socket Layer authentication. To increase the degree of abstraction at the toolkit interface, we are moving towards the use of the Generic Security System (GSS) [13]. GSS defines a standard procedure and API for obtaining credentials (passwords or certificates), for mutual authentication (client and server), and for message-oriented encryption and decryption. GSS is independent of any particular security mechanism and can be layered on top of different security methods, such as Kerberos and SSL.

GSS must be altered and extended to meet the requirements of metacomputing environments. As a metacomputing system may use different authentication mechanisms in different situations and for different purposes, we require a GSS implementation that supports the concurrent use of different security mechanisms. In addition, inquiry and selection functions are needed so that higher-level services implementing specific security policies can select from available low-level security mechanisms. The Globus authentication interface can be used to implement a range of different security policies. We are currently investigating a policy that defines a global, public key-based authentication space for all users and resources. That is, we provide a centralized authority that defines system-wide names ("accounts") for users and resources. These names allow an application to use a single "user id" and "password" for all resources. They also permit the application to verify the identity of requested resources. Note that this policy does not address authorization: resources can use their usual mechanisms to determine the users to which they will grant access.

While not practical for large-scale, open environments, the use of a centralized authority to identify users and resources is appropriate for limited-scale testbed environments such as the I-WAY and GUSTO (see below). The approach has the significant advantage that it can be implemented easily with current certificate-based authentication protocols, such as that provided in the Secure Socket Library (SSL). Note that while names (certificates) are issued by a centralized certificate authority, the authentication of users and services involves only the agents being authenticated; it does not require any interaction with the issuing authority.

In the longer term, authentication and authorization schemes must address the requirements of larger, dynamic, heterogeneous communities, in which trust relationships span multiple administrative domains and can be irregular and selective. Some member organizations will be more trusting of specific members than others; still others may be competitors. Some members may feel the need to control all trust relationships explicitly, even if it means that fewer community assets are available for their use; this may stimulate the evolution of virtual communities with their own set of trust and authorization relationships. Community members willing to delegate to other members the ability to extend relationships on their behalf may more fully enjoy the benefits of membership in the greater community. In the long run, membership in the community is likely to be bolstered by an economy-of-scale argument which will be a direct consequence of sharing and trust.

Our certificate-based policy can be extended to support limited forms of trust delegation. Globus resource certificates can be given to sites with multiple resources (or users). These sites can in turn set up a local certificate authority, which signs certificates that it issues with the certificate issued by the Globus authority. This situation is acceptable if the user (or resource) trusts the administration of the site issuing the Globus-signed certificate.

A certificate based authentication mechanism seems to be sufficient for a telemedical infrastructure. Each domain will handle its certificates to allow only access to site or application relevant services. This authentication can also be used to allow secure transmission on data via communication links.

3.5 Data Access Services

Services that provide metacomputing applications with access to persistent data can face stringent performance requirements and must support access to data located in multiple administrative domains. Distributed file systems such as the Network File System and Distributed File System address remote accesses to some extent but have not been designed for high-performance applications. Parallel file systems and I/O libraries have been designed for performance but not for distributed execution.

To address these problems, the Globus data access module defines primitives that provide remote access to parallel file systems. This remote I/O (RIO) interface (Figure 5) is based on the abstract I/O device (ADIO) interface [28]. ADIO defines an interface for opening, closing, reading and writing parallel files. It does not define semantics for caching, file replication, or parallel file descriptor semantics. Several popular I/O systems have been implemented efficiently on ADIO [28]. RIO extends ADIO by adding transparent remote access and global naming using a URL-based naming scheme.

Together with a directory service as previously introduced the data access can be achieved uniformly over a large number of compute and data resources.

Even though for the programmer the direct file access if of particular interest, it is more essential for an applications programmer to locate the “object” patient, MRI, etc. This leads us to the need to introduce high level services, which can be provided while reusing the above introduced low level services. Again, as addressed earlier in the paper, access to patient data can be done securely. Differing views of patient data can be exposed via a hybrid design which incorporates LDAP and relational or object database technologies.

3.5.1 Parallel Programming Interfaces

Numerous higher level parallel programming interfaces 4 have been adapted to use Globus authentication, process creation, and communication services, hence allowing programmers to develop metacomputing applications using familiar tools. These interfaces include a complete implementation of MPI (and hence tools layered on top of MPI, such as many High Performance Fortran systems); Compositional C++ [2], a parallel extension to C++; Fortran M, a task-parallel Fortran; nPerl, a version of the Perl scripting language extended with remote reference and remote procedure call mechanisms; and NexusJava, a Java class library that supports remote procedure calls between Java and other Nexus-enabled components. These higher-level programming interfaces are of special importance since they will be used in the application implementation of a telemedical infrastructure. Due to the heterogeneity of the many telemedical applications, it is important to support a wide range of higher level programming interfaces. The availability of proper high level programming interfaces will ultimately reduce the development costs of the telemedical infrastructure. One very important property about the Globus environment is stressed here to fully demonstrate the usefulness of the Globus toolkit components: The higher level programming modules all use the Nexus communication library. This makes it possible to develop truly heterogeneous programs, which use different higher-level parallel programming interfaces. While looking back to the scenario introduced at the beginning of the paper, such a truly heterogeneous application could use a Fortran based program for weather prediction and use the Nexus communication routines to interface with a traffic control simulation system running on a supercomputer in CC++. Furthermore, the data collected in the ambulance with specialized instruments could be transferred to the hospital which displays this information with a Java based GUI.

3.6 Resource Scheduling

Many telemedical applications make use of expensive equipment. Scheduling and reservation algorithms and policies have to be developed in order to make maximal use of the infrastructure. Not only the instruments have to be scheduled and reserved, but also the computational and knowledge resources in order to, make a real time usage possible. We can distinguish scheduled and unscheduled modes of operation. In *scheduled* modes, resources, once acquired, are dedicated to an application. In *unscheduled* mode, applications use otherwise idle resources that may be reclaimed if needed; Condor [14] is one system that supports this mode of operation. In general, scheduled mode is required for tightly coupled simulations, particularly those with time constraints, while unscheduled mode is appropriate for loosely coupled applications that can adapt to time-varying resources. A possible example to use such an environment is the distributed solution of gene sequencing problems which will have increased importance in the near future[9]. The resources to be scheduled include desktop supercomputers, smart instruments, collaborative environments [3,4], and distributed supercomputers [16,18,19].

Telemedical applications often need to operate networking and computing resources at close to maximum performance. Hence, metacomputing environments for telemedical applications must allow programmers to observe differences in system resource characteristics and to guide how these resources are used to implement higher-level services. Achieving these goals without compromising portability is a significant challenge for the designer of metacomputing software.

We use the Globus communication module to illustrate some of these issues. This module must select, for each call to its communication functions, one of several low-level mechanisms. On a local area network, communication might be performed with TCP/IP, while in a parallel computer, specialized high-performance protocols typically offer higher bandwidth and lower latencies. In a wide area environment, specialized ATM protocols can be more efficient. The ability to manage protocol parameters (TCP packet size, network quality of service) further complicates the picture. The choice of low-level mechanism used for a particular communication is a nontrivial problem that can have significant implications for application performance.

Globus toolkit modules address this problem by providing interfaces that allow the selection process to be exposed to, and guided by, higher-level tools and applications. These interfaces provide rule-based *selection*, resource property *inquiry*, and *notification* mechanisms.

- *Rule-based selection.* Globus modules can identify selection points at which choices from among alternatives (resources, parameter values, etc.) are made. Associated with each selection point is a default selection rule provided by the module developer (e.g., "use TCP packet size X," "use TCP over ATM"). A rule replacement mechanism allows higher-level services to specify alternative strategies ("use TCP packet size Y," "use specialized ATM protocols").

- *Resource property inquiry.* Information provided by the unified information service can be used to guide selection processes within both Globus modules and applications that use these modules. For example, a user might provide a rule that states "use ATM interface if load is low, otherwise Internet," hence using information about network load to guide resource selection.
- *Notification.* A notification mechanism allows a higher-level service or application to specify constraints on the quality of service delivered by a Globus service and to name a call-back function that should be invoked if these constraints are violated. This mechanism can be used, for example, to switch between networks when one becomes loaded.

Higher-level services and applications can use Globus selection, inquiry, and notification mechanisms to *configure* computations efficiently for available resources, and/or to *adapt* behavior when the quantity and/or quality of available resources changes dynamically during execution. For example, consider an application that performs computation on one computer and transfers data over a wide area network for visualization at remote sites. At startup time, the application can determine available computational power and network capacity and configure its computational and communication structures appropriately (e.g., it might decide to use compression for some data but not others). During execution, notification mechanisms allow it to adapt to changes in network quality of service (see figures 6 and 7).

4 Future opportunities

Having demonstrated a metacomputing environment in a real testbed, new research opportunities will arise. Telemedicine is just one of them. We would like to emphasize that telemedicine is an ideal application to test the principles of a metacomputer due to its inherent heterogeneous characteristics. Assuming the compute and network power is in place one will ask, what is the next step?

We predict that the accuracy of a medical exam is likely to improve. We also believe that research areas that are dependent on many calculations or a huge number of data will have an increased impact on treatments. Furthermore, we will see an increased number of interdisciplinary research enhance our knowledge about medicine. Examples will include for example sophisticated observation networks which warn patients weather conditions, home monitoring of a patients health progress with wireless technology, the integration of immense amounts of data in research databases. Furthermore, a metacomputing environment as defined with Globus toolkit components will allow specifying various access policies of data in the highly controversial issue of a global database. Each information provider will be able to establish its own information policy.

5 Conclusion

The Globus project is attacking the metacomputing software problem from the bottom up, by developing basic mechanisms that can be used to implement a variety of higher-level services. Communication, resource location, resource allocation, information, authentication, data access, and other services have been identified, and considerable progress has been made toward constructing quality implementations. The definition, development, application, evaluation, and refinement of these components are ongoing processes that we expect to proceed for the next two years at least. We hope to involve more of the metacomputing community in this process, by adapting relevant higher-level services (e.g., application-level scheduling [1], performance steering [22], object-based libraries [11]) to use Globus mechanisms, and by participating in the construction of additional testbeds (e.g., GUSTO).

The Globus project is also addressing the configuration problem in metacomputing systems, with the goal of producing an Adaptive Wide Area Resource Environment that supports the construction of adaptive services and applications. We have introduced selection, information, and notification mechanisms and have defined Globus component interfaces so that these mechanisms can be used to guide the configuration process. Preliminary experiments with dynamic communication selection suggest that these configuration mechanisms can have considerable value [6].

In summary, we list three areas in which we believe the Globus project has already made contributions and in which we hope to see considerable further progress:

- The definition of a core metacomputing system architecture on which a range of alternative metacomputing environments can be built.
- The development of a framework that allows applications to respond to dynamic behaviors in the underlying metacomputing environment, and the definition and evaluation of various adaptation policies.
- The demonstration in testbeds such as the I-WAY and GUSTO that useful higher-level services can be layered effectively on top of the interfaces defined by the Globus toolkit, and that automatic configuration mechanisms can be used to enhance portability and performance.

Throughout paper we showed examples how to use the different low level and high level services to support a telemedical infrastructure. Since the characteristics of a telemedical environment are almost identical to a metacomputing environment, the Globus toolkit components can be reused to build a telemedical environment. In addition, the Globus toolkit shows how novel approaches, e.g., the use of directory services can be extended to represent patient data in a distributed compute environment. The authentication mechanism with keys can serve as an example how to enable secure communication between different administrative domains. Compatibility is assured due to the availability of the Globus toolkit components on a variety of platforms and the availability of a Java based implementation. Due to the modular characteristic of the toolkit it can be extended which is important for a changing infrastructure in a telemedical environment.

6 Acknowledgements

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. Globus research and development is supported by DARPA, DOE, and NSF.

References

[1] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, *Application-level scheduling on distributed heterogeneous networks*, in Proceedings of Supercomputing '96, ACM Press, 1996.

[2] K. M. Chandy and C. Kesselman, *CC++: A declarative concurrent object oriented programming notation*, in Research Directions in Object Oriented Programming, The MIT Press, 1993, pp. 281–313.

[3] D. Diachin, L. Freitag, D. Heath, J. Herzog, W. Michels, and P. Plassmann, *Remote engineering tools for the design of pollution control systems for commercial boilers*, International Journal of Supercomputer Applications, 10 (1996), pp. 208–218.

[4] T. L. Disz, M. E. Papka, M. Pellegrino, and R. Stevens, *Sharing visualization experiences among remote virtual environments*, in International Workshop on High Performance Computing for Computer Graphics and Visualization, Springer-Verlag, 1995, pp. 217–237.

[5] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, *A directory service for configuring high-performance distributed computations*, in Proc. 6th IEEE Symp. on High Performance Distributed Computing, IEEE Computer Society Press, 1997, pp. 365–375.

[6] I. Foster, J. Geisler, C. Kesselman, and S. Tuecke, *Managing multiple communication methods in high-performance networked computing systems*, Journal of Parallel and Distributed Computing, 40 (1997), pp. 35–48.

[7] I. Foster, N. Karonis, C. Kesselman, G. Koenig, and S. Tuecke, *A secure communications infrastructure for high-performance distributed computing*, in Proc. 6th IEEE Symp. on High Performance Distributed Computing, IEEE Computer Society Press, 1997, pp. 125–136.

[8] I. Foster, C. Kesselman, and S. Tuecke, *The Nexus approach to integrating multithreading and communication*, Journal of Parallel and Distributed Computing, 37 (1996), pp. 70–82.

[9] T. Gasterland, *Magpie system.* http://www.mcs.anl.gov/home/gaasterl/papers.html.

[10] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine—A User's Guide and Tutorial for Network Parallel Computing*, MIT Press, 1994.

[11] A. Grimshaw and W. Wulf, *Legion – a view from 50,000 feet*, in Proc. 5th IEEE Symp. on High Performance Distributed Computing, IEEE Computer Society Press, 1996, pp. 89–99.

[12] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message Passing Interface*, The MIT Press, 1994.

[13] J. Linn, *Generic security service application program interface*, Internet RFC 1508, (1993).

[14] M. Litzkow, M. Livney, and M. Mutka, *Condor - a hunter of idle workstations*, in Proc. 8th Intl Conf. on Distributed Computing Systems, 1988, pp. 104–111.

[15] A. Mainwaring, *Active Message applications programming interface and communication subsystem organization*, tech. rep., Dept. of Computer Science, UC Berkeley, Berkeley, CA, 1996.

[16] C. Mechoso et al., *Distribution of a Coupled-ocean General Circulation Model across high-speed networks*, in Proceedings of the 4th International Symposium on Computational Fluid Dynamics, 1991.

[17] J. Morris, M. Satyanarayanan, M. Conner, J. Howard, D. Rosenthal, and F. Smith, *Andrew: A distributed personal computing environment*, Communications of the ACM, 29 (1986), pp. 184–201.

[18] J. Nieplocha and R. Harrison, *Shared memory NUMA programming on the I-WAY*, in Proc. 5th IEEE Symp. on High Performance Distributed Computing, IEEE Computer Society Press, 1996, pp. 432–441.

[19] M. Norman, P. Beckman, G. Bryan, J. Dubinski, D. Gannon, L. Hernquist, K. Keahey, J. Ostriker, J. Shalf, J. Welling, and S. Yang, *Galaxies collide on the I-WAY: An example of heterogeneous wide-area collaborative supercomputing*, International Journal of Supercomputer Applications, 10 (1996), pp. 131–140.

[20] U. C. O. of Technology Assessment, *Bringing health care online: the role of information technologies.* Washington, DC: U.S. Government Printing Office, OTA-ITC-624, Sept. 95.

[21] S. Pakin, M. Lauria, and A. Chien, *High performance messaging on workstations: Illinois Fast Messages (FM) for Myrinet*, in Proceedings of Supercomputing '95, IEEE Computer Society Press, 1996.

[22] D. Reed, C. Elford, T. Madhyastha, E. Smirni, and S. Lamm, *The Next Frontier: Interactive and Closed Loop Performance Steering*, in Proceedings of the 1996 ICPP Workshop on Challenges for Parallel Processing, Aug. 1996, pp. 20–31.

[23] *Telemedicine and advanced technology research center(tatrc).* http://www.matmo.org.

[24] *Links to telemedinine on the net.* http://www.jma.com.au/telelink.htm.

[25] *Links to telemedinine on the net.* http://www.yahoo.com/Health/Medicine/Informatics/Telemedicine/.

[26] *Telemedicine information exchange.* http://tie.telemed.org.

[27] *Telemedicine report to congress.* http://www.ntia.doc.gov/reports/telemed/index.htm, Jan. 97. A report on the use of advanced telecommunications services for medical purposes, that also examines questions relating to efficacy,safety and quality of services, together with other legal, medical and economic issues.

[28] R. Thakur, W. Gropp, and E. Lusk, *An abstract-device interface for implementing portable parallel-I/O interfaces*, in Proceedings of The 6th Symposium on the Frontiers of Massively Parallel Computation, October 1996.

[29] G. von Laszewski and I. Foster, *Supercomputing in structural biology.* htttp://www.mcs.anl.gov/xray.

[30] R. Wolski, *Dynamically forecasting network performance using the network weather service*, Tech. Rep. TR-CS96-494, U.C. San Diego, October 1996.

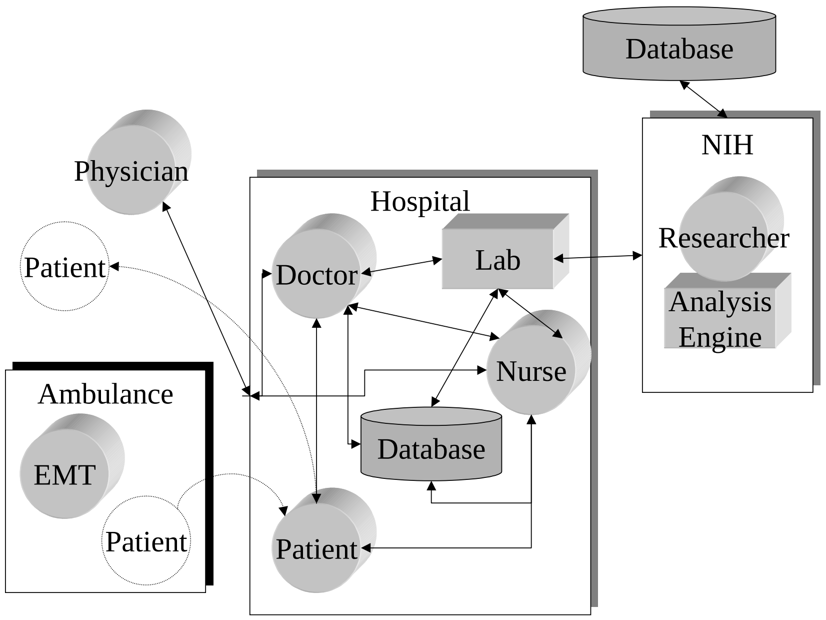


Fig. 1. An example scenario for a telemedical infrastructure

Table 1
Storage requirements for imaging techniques and transmission times while using different network transport medias/services

					Phone	ISDN	T1	T3	ADSL	ATM
pixels	image	average number	calc	average storage	28.8	128	1.54	45.3	1.5	155
hor vert	depth	of images/exam	memory	requirement	Kbits/s	Kbits/s	Mbytes/s	Mbytes/s	Mbytes/s	Mbytes/s
512	12	30	12	15	69.4	15.6	9.7	0.3	10.0	0.10
256	12	50	3	6.5	30.1	6.8	4.2	0.1	4.3	0.04
1000	8	20	45	20	92.6	20.8	13.0	0.4	13.3	0.13
1000	8	15	45	15	69.4	15.6	9.7	0.3	10.0	0.10
512	6	36	12	9	41.7	9.4	5.8	0.2	6.0	0.06
128	8	26	1	0.4	1.9	0.4	0.3	0.01	0.3	0.003
2000	10	4	180	32	148.1	33.3	20.8	0.7	21.3	0.21
4000	12	4	720	128	592.6	133.3	83.1	2.8	85.3	0.83
					in min	in min	in sec	in sec	in sec	in sec

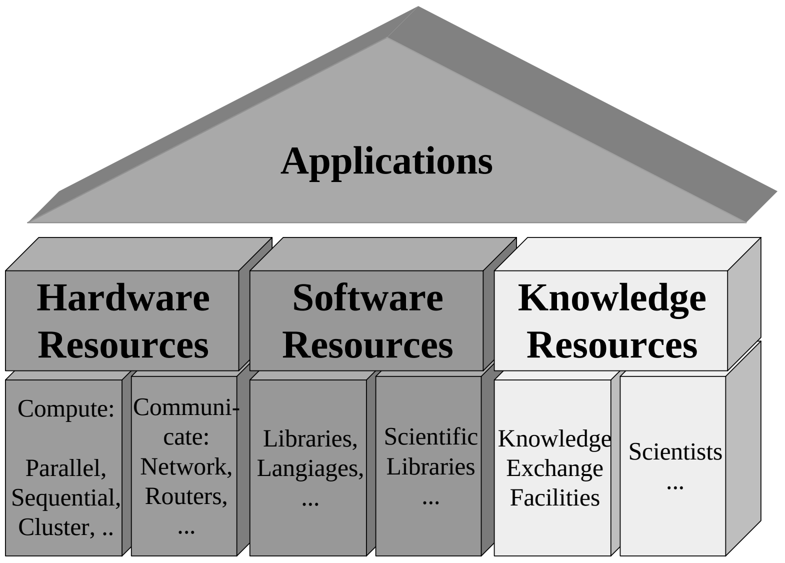

Applications
Hardware Resources
Software Resources
Knowledge Resources
Compute: Parallel, Sequential, Cluster, ..
Communi-cate: Network, Routers, ...
Libraries, Langiages, ...
Scientific Libraries ...
Knowledge Exchange Facilities
Scientists ...

Fig. 2. Resources needed to implement applications in a metacomputing environment

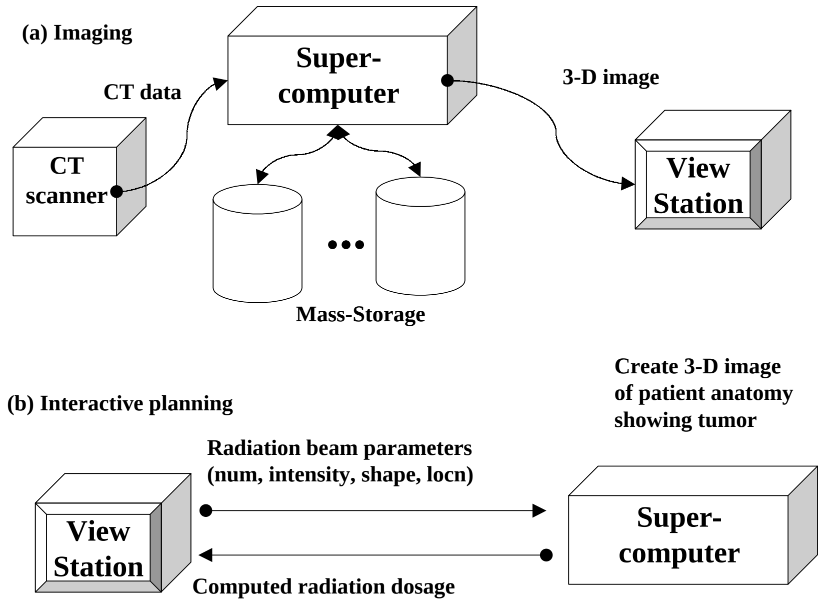

(a) Imaging
CT data
Super-computer
3-D image
CT scanner
View Station
Mass-Storage
(b) Interactive planning
Create 3-D image of patient anatomy showing tumor
Radiation beam parameters (num, intensity, shape, locn)
View Station
Super-computer
Computed radiation dosage

Fig. 3. Usage of supercomputers in imaging and computational steering

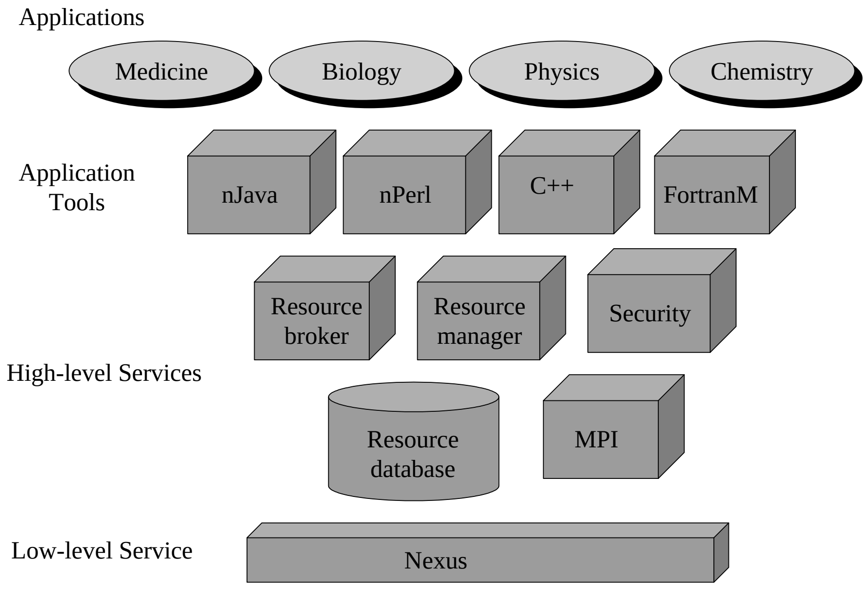


Fig. 4. Components of the Globus toolkit

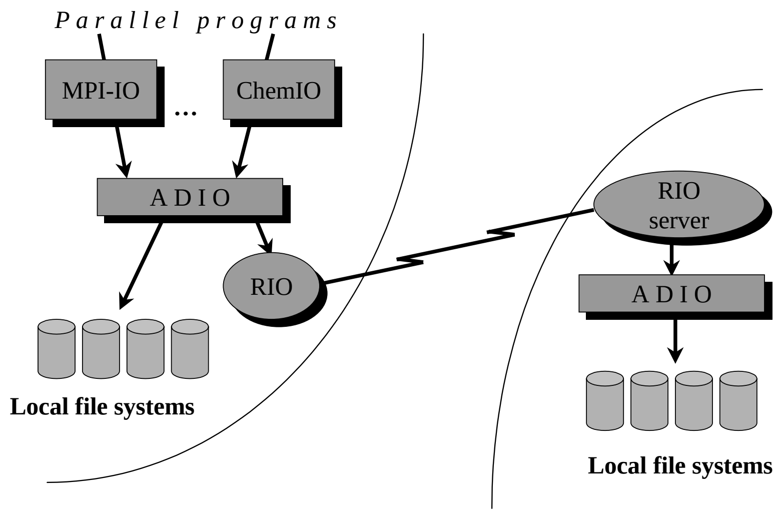


Fig. 5. Remote IO mechanisms for high-performance access to remote file systems

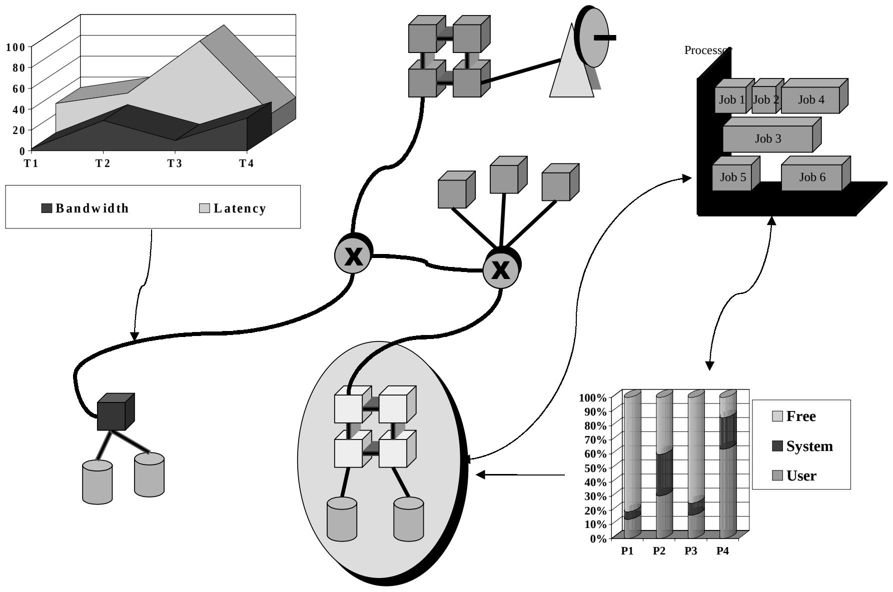
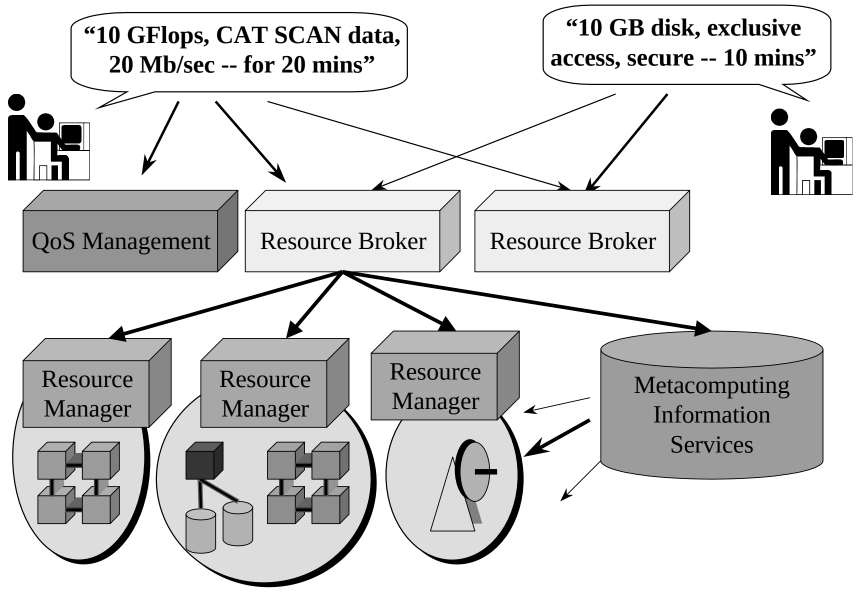


Fig. 6. A steering environment based on a metacomputing framework



"10 GFlops, CAT SCAN data, 20 Mb/sec -- for 20 mins"
"10 GB disk, exclusive access, secure -- 10 mins"
QoS Management
Resource Broker
Resource Broker
Resource Manager
Resource Manager
Resource Manager
Metacomputing Information Services

Fig. 7. Resource selection and monitoring